Improving SLAM with Drift Integration

Guillaume Bresson, Romuald Aufrère, Roland Chapuis

Improving SLAM with drift integration

Guillaume Bresson[1,2], Romuald Aufrère[3,4] and Roland Chapuis[3]

***Abstract*— Localization without prior knowledge can be a difficult task for a vehicle. An answer to this problematic lies in the Simultaneous Localization And Mapping (SLAM) approach where a map of the surroundings is built while simultaneously being used for localization purposes. However, SLAM algorithms tend to drift over time, making the localization inconsistent. In this paper, we propose to model the drift as a localization bias and to integrate it in a general architecture. The latter allows any feature-based SLAM algorithm to be used while taking advantage of the drift integration. Based on previous works, we extend the bias concept and propose a new architecture which drastically improves the performance of our method, both in terms of computational power and memory required. We validate this framework on real data with different scenarios. We show that taking into account the drift allows us to maintain consistency and improve the localization accuracy with almost no additional cost.**

I. Introduction

In the quest for making vehicles autonomous, the scientific community revolving around mobile robotics has furnished an impressive amount of work towards localization without any prior knowledge. A vehicle must be able to locate itself accurately in its environment so as to take the right decisions (choose which path to follow for instance) in total autonomy. While some could think of GPS as the solution to this localization problem, it has quickly emerged that it was not enough, both in terms of accuracy (or cost needed to reach a proper accuracy with devices like RTK-GPS's) and reliability (urban canyons, etc.). The topic of Simultaneous Localization And Mapping (SLAM) has thus become part of the key to this localization issue.

While many SLAM solutions, relying on various sensors and methods, have been proven to work, some aspects are often overlooked, such as how to properly handle the inherent SLAM drift or how to have a framework which could work with different settings (as proposed in [11]) and deal with high-level issues (detect loops, integrate geo-referenced information, take into account the localization drift, etc.).

The localization provided by SLAM algorithms tend to drift over time [2]. Improving the localization capabilities is thus tightly linked to the integration of all sorts of information that can help reduce this drift. We can divide them in three categories that overlap. Geo-referenced data is a first way to counter the drift as it is not affected by it [7]. Finding loops in the trajectory (recognizing previously visited places) is another way. By identifying previously perceived landmarks, it becomes possible to reduce the drift to what it was when landmarks were first seen [12]. Last, information coming from external agents (other vehicles, infrastructure, etc.) can at least constrain the drift as they are not similarly (or at all) affected by it [3]. In order to properly integrate this knowledge in a SLAM algorithm, a global sensor independent architecture should be prefered.

In this article, we propose a solution to solve both problems with a drift model integrated inside a dedicated architecture. Based on previous works [3], we improve the overall performance of our system with a new drift integration and a different handling of the map state. We present here an application to the loop closing problem using a data association process which is not sensor dependent as well as an example of geo-referenced integration.

We will first present related work regarding SLAM drift in Section II. Then, in Section III, we will give further details about the drift model that we previously introduced in [3]. Next, in Section IV, we will focus on the new architecture proposed in this paper to deal with the drift problem. We will expose the integration of a feature-based SLAM and how the drift is taken into account. Finally, an experimental validation will be provided in Section V along with various metrics and resource consumption comparisons.

II. Related work

The drift is causing the computed localization (and therefore the map as well) to slowly diverge from its true value. Moreover, the uncertainty associated to the vehicle pose does not include the real pose. It means that the computed estimate is inconsistent and should not be trusted.

The divergence problem affecting SLAM algorithms has been stated several times in the literature. The use of non-linear models that need to be linearized is one of the reason causing the divergences. Indeed, linearizations made far from the true value tend to produce optimistic estimates [2]. The consequence is that the localization slowly drifts over time. Of course, the use of any sensor from whose errors accumulate over time (odometer for instance) will also provoke this phenomenon. Correlations between sensor data are also a source of overconfidence [10]. Indeed, most of the time, it is wrongly considered that data coming from the same sensor are not temporally linked. It also appears that an approximation regarding the vehicle orientation has a stronger impact on the divergence than a translation error [1]. The literature regarding SLAM drift usually considers two ways of dealing with it: by avoiding it or by reducing it when possible.

[1]Institut VEDECOM, Versailles, France `firstname.name@vedecom.fr`

[2]Inria Paris-Rocquencourt, Le Chesnay, France

[3]Institut Pascal - UMR 6602 CNRS, Université Blaise Pascal - Aubière, France `firstname.name@univ-bpclermont.fr`

[4]LIMOS - UMR 6158 CNRS, Université Blaise Pascal - Aubière, France

The former category includes submapping approaches [9] where new reference frames are regularly set. At a submap level, most of the drift is avoided. It can be an issue as creating too much submaps leads to an information loss. However, the global map linking all the submaps is still affected by the divergence. In a similar manner, robot-centric methods are often considered to avoid drift [5]. Estimates are always given with relation to the robot pose thus limiting consistency problems as they do not accumulate over time. However, as for submapping techniques, divergence is not solved when considered out of the robot-centered frame which is needed to detect loop closures and in many other applications. Some authors have worked on reducing errors with smoothing approaches [8]. Even with such methods, the remaining drift could still be taken into account to improve consistency.

Regarding reducing drift, in [7], satellite images are used to obtain an absolute localization. The satellite images are compared to the output of a standard camera. The vehicle pose can be computed based on these absolute, and so drift-free, correspondences. A geo-referenced database is also a good way to remove the drift and maintain consistency. However, most of the time, absolute information can only be integrated punctually and there is no guarantee that the system will not drift in-between. In [12], the authors show that even if closing the loop reduces the drift, the results will always be overconfident because of approximations regarding the links between landmarks. Cole and Newman in [6] propose to redistribute errors in a probabilistic manner around the past trajectory. Even if better results are obtained, the resulting map is not always consistent. Besides, the vehicle pose will certainly be diverging without knowing so before finding any loop closing.

The solutions presented in this last paragraph can all be applied to reduce the drift. However, it appears mandatory to be able to represent the drift at any time even without such information. This is why we propose here to model the drift and to integrate it inside a Kalman filter so as to estimate it conjointly with the vehicle state and the map.

III. Drift

We define the localization bias as the parameters allowing to compensate the drift. As the drift tends to grow with the distance traveled, the localization bias evolves according to the curvilinear abscissa. As so, at a curvilinear abscissa s, the bias is defined as:

$$\mathbf{b}_s = \begin{pmatrix} b_{x_s} & b_{y_s} & b_{\theta_s} \end{pmatrix}^T \tag{1}$$

with b_{x_s} and b_{y_s} representing a position bias and b_{θ_s} an angular bias.

The localization bias follows a dynamic model. Indeed, the bias at a curvilinear abscissa s strongly depends on the one preceding it at $s - \Delta s$. Starting from the evolution equation of a continuous stochastic linear system [15], we can write:

$$\dot{\mathbf{b}}(s) = \mathbf{A}_c \mathbf{b}(s) + \mathbf{B}_c \mathbf{u}(s) + \mathbf{M}_c \boldsymbol{\varepsilon}(s) \tag{2}$$

where $\mathbf{b}$ is the bias vector, $\mathbf{u}$ the input vector and $\boldsymbol{\varepsilon}$ the evolution noise which is assumed white. $\mathbf{A}_c$ is the state matrix associated to $\mathbf{b}$, $\mathbf{B}_c$ expresses the link between $\mathbf{u}$ and $\mathbf{b}$ and $\mathbf{M}_c$ the link between $\boldsymbol{\varepsilon}$ and $\mathbf{b}$.

The drift is considered to evolve similarly across its three parameters and as such, a simple random walk can be used as an approximate model. Considering so, $\mathbf{A}_c = 0$, $\mathbf{B}_c = 0$ and $\mathbf{M}_c = \mathbf{I}$ (identity matrix), and the drift evolution becomes:

$$\dot{\mathbf{b}}(s) = \boldsymbol{\varepsilon}(s) \tag{3}$$

By discretizing Equation (3), we obtain the following bias evolution:

$$\begin{aligned} \mathbf{b}_s &= \mathbf{A}\mathbf{b}_{s-\Delta s} + \boldsymbol{\varepsilon}_{\Delta s} \\ &= \exp(\mathbf{A}_c \Delta s)\mathbf{b}_{s-\Delta s} + \boldsymbol{\varepsilon}_{\Delta s} \\ &= \mathbf{b}_{s-\Delta s} + \boldsymbol{\varepsilon}_{\Delta s} \end{aligned} \tag{4}$$

with $\boldsymbol{\varepsilon}_{\Delta s}$ being the drift (white noise) affecting the vehicle between $s - \Delta s$ and s.

In order to ensure consistency at any time, the uncertainty regarding the drift occurring between $s - \Delta s$ and s must be considered. It is expressed as follows:

$$\mathbf{P}_{\varepsilon_{\Delta s}} = \int_0^{\Delta s} \exp(\mathbf{A}_c s)\mathbf{P}_{\varepsilon} \exp(\mathbf{A}_c^T s) ds \tag{5}$$

with $\mathbf{P}_{\varepsilon}$ being the uncertainty associated to $\boldsymbol{\varepsilon}(s)$.

Based on Equations (4) and (5), the bias uncertainty $\mathbf{P}_{\mathbf{b}_s}$ is calculated as:

$$\mathbf{P}_{\mathbf{b}_s} = \mathbf{P}_{\mathbf{b}_{s-\Delta s}} + \Delta s \mathbf{P}_{\varepsilon} \tag{6}$$

We consider here a stationary process. Even though the bias evolution is not identical in every situation (for instance, turning might have a stronger impact on the rotation drift than going straight), this approximation is sufficient if $\mathbf{P}_{\varepsilon}$ is properly characterized. As a consequence of the stationarity hypothesis, $\mathbf{P}_{\varepsilon}$ can be experimentally defined, based on the observed quadratic divergence over the distance traveled.

Considering the vehicle pose $\mathbf{v}_k = \begin{pmatrix} x_k & y_k & \theta_k \end{pmatrix}^T$ at a time k corresponding to the curvilinear abscissa s, the localization bias can be integrated as follows (where u stands for unbiased):

$$\begin{aligned} \mathbf{v}_{u_k} &= \begin{bmatrix} \mathbf{R}(b_{\theta_s}) & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} \mathbf{v}_k + \mathbf{b}_s \\ &= \begin{bmatrix} \cos(b_{\theta_s}) & -\sin(b_{\theta_s}) & 0 \\ \sin(b_{\theta_s}) & \cos(b_{\theta_s}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v}_k + \mathbf{b}_s \end{aligned} \tag{7}$$

Similarly, for a landmark $\mathbf{l}_{i_k} = \begin{pmatrix} l_{i_{x_k}} & l_{i_{y_k}} \end{pmatrix}^T$, the localization bias can be taken into account:

$$\mathbf{l}_{u_{i_k}} = \mathbf{R}(b_{\theta_s})\mathbf{l}_{i_k} + \begin{bmatrix} b_{x_s} \\ b_{y_s} \end{bmatrix} \tag{8}$$

Based on Equations (7) and (8), we can also infer how the uncertainty regarding the vehicle pose and a landmark

position will be affected by the drift integration. However, integrating the bias localization directly into the SLAM process where these estimates are computed will not prevent the system from being inconsistent. Indeed, observations will lower the bias uncertainty while it should not be the case. A dedicated architecture is thus needed to avoid this phenomenon.

IV. Global architecture

The objective of this architecture is twofold: providing a framework in which any feature-based SLAM algorithm can take into account and handle the drift as well as integrating all the possible means to estimate it thanks to the ensured consistency. To do so, the initial SLAM should be decorrelated from the part responsible for the drift estimation. It has the advantage to avoid mixing uncertainties and to allow the use of any feature-based SLAM algorithm. This led us to the organization presented in Figure 1.

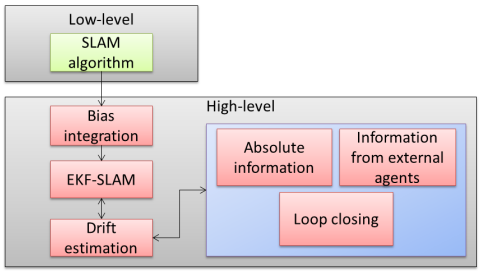


Fig. 1. Bias integration architecture

It can be noted that the low-level SLAM algorithm just communicates with the rest of the architecture by providing landmark and vehicle estimates. A dedicated process takes these incoming landmarks and vehicle poses and builds a proper map, designed to be consistent due to the bias integration. Specific modules, whose goal is the drift estimation, can then be developed (squared in blue in Figure 1) by taking advantage of this new consistent map. Each module provides a way to estimate the bias. The drift estimation module is in charge of activating them and provides the information to fuse to the Extended Kalman Filter (EKF). So as to balance the computing requirements, landmarks do not need to be kept in the low-level algorithm when they are not visible anymore. As associations (loops for instance) are sought in the high-level, it is not useful to have all the landmarks in the low-level (and thus allowing the use of Visual Odometry algorithms). This way, the architecture does not add much to a classic SLAM in terms of memory or processing power required.

An EKF is used to handle the high-level map. Its goal is to connect the bias estimates together based on Equation (4) and update them when an information allowing so is available (loop closing, geo-referenced data, etc.). Bias estimates are regularly inserted (by the Bias integration module) based on the distance traveled in order to represent the drift along the trajectory. While there should ideally be a bias estimate per landmark, it would require an important amount of resources to handle them and would not improve much the localization quality. With no prior knowledge about the drift ($\varepsilon = 0$), its initialization via the Kalman Filter is simple:

$$\mathbf{b}_s = \mathbf{b}_{s-\Delta s} \tag{9}$$

In order to connect the current bias to its previous estimate, we first initialize its variance to infinite values inside the covariance matrix and use the EKF to refine it. To do so, let define the observation function h_b which makes the difference between two consecutive bias estimates:

$$h_b(\mathbf{b}_s, \mathbf{b}_{s-\Delta s}) = \mathbf{b}_s - \mathbf{b}_{s-\Delta s} \tag{10}$$

We define the observation error as follows:

$$\mathbf{R}_b = \mathbf{P}_{\mathbf{b}s} - \mathbf{P}_{\mathbf{b}s-\Delta s} \tag{11}$$

$\mathbf{H}_b$, the Jacobian matrix associated to h_b, can be computed:

$$\mathbf{H}_b = \begin{bmatrix} \mathbf{0}_{3\times 3(n-2)} & -\mathbf{I}_{3\times 3} & \mathbf{I}_{3\times 3} \end{bmatrix} \tag{12}$$

where n is the number of bias estimates in the state vector.

A Kalman update is then performed to link together bias estimates and properly initialize the uncertainty of the new bias estimate $\mathbf{b}_s$. The lack of knowledge about ε makes the Kalman innovation equal to zero, meaning that only the covariance matrix will be affected by the update.

Instead of inserting landmarks and vehicle poses into the state vector, we chose to only have connected bias estimates. The main motivation is to reduce the memory used by the storage of a rapidly growing covariance matrix as well as the processing power needed to properly insert a new element inside the map. Indeed, the EKF requires several multiplications involving the whole state vector and covariance matrix and, even though it is not an important burden with a few landmarks, it quickly becomes unbearable with hundreds of them. As high-level landmarks and vehicle poses only depend on the bias estimation, there is no loss of information. The only constraint comes from the data association process which will be later discussed in this section. The high-level architecture is depicted in Figure 2.

Incoming landmarks are stored along with the current vehicle pose and the respective variances (1). The cross-covariances recomposed thanks to the previously presented drift integration are indicated in blue here (2). The vehicle state vector taking into account the bias (3) is here to help the data association process.

In order to correct the drift, landmarks must be associated and fused. The three modules presented earlier in Figure 1 work similarly. They consider incoming landmarks and look for associations across landmarks already in the high-level (stored in (1) in Figure 2). The difference lies in the fact that the fusion does not affect the same bias estimates. In case of information coming from another vehicle for instance, the bias estimates of both vehicles involved are updated when an association is found. For geo-referenced

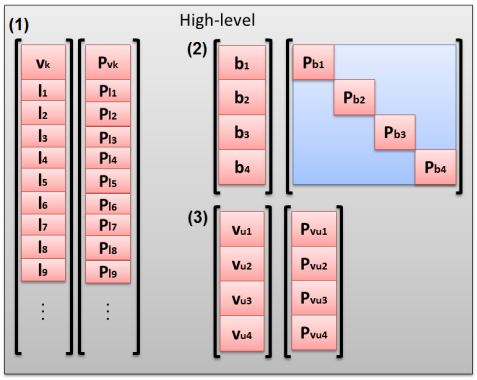


Fig. 2. High-level architecture

information, the bias estimates of the vehicle with which information is fused are concerned. Regarding loop closing, the fusion occurs between landmarks already mapped and new ones, which means that the bias estimates regarding these two sets of landmarks become directly connected. In each case, the cross-covariances recomposed (2) are able to properly spread the impact of an update on all the bias estimates concerned.

Let consider that a new landmark $\mathbf{l}_j$ (associated to a bias $\mathbf{b}_j$) corresponds to a landmark already mapped $\mathbf{l}_i$, the observation $\mathbf{z} = \mathbf{l}_j$ and its covariance $\mathbf{R} = \mathbf{P}_{\mathbf{l}_j}$ as received from the low-level SLAM (or a distant vehicle, or already known geo-referenced information). The association found means that $\mathbf{l}_{u_j} = \mathbf{l}_{u_i}$ which allows us to define the non-linear observation function h_f as:

$$h_f(\mathbf{b}_i, \mathbf{b}_j) = \mathbf{R}(b_{\theta_j})^T \left(\mathbf{R}(b_{\theta_i})\mathbf{l}_i + \begin{bmatrix} b_{x_i} \\ b_{y_i} \end{bmatrix} - \begin{bmatrix} b_{x_j} \\ b_{y_j} \end{bmatrix} \right) \quad (13)$$

The Jacobian $\mathbf{H}_f$ associated to the function h_f is then computed and used in the Kalman equations to update the state vector composed of bias estimates. All the bias estimates will be affected by the update. Of course, the farthest from the fusion point, the less concerned by the update will be bias estimates. By only working with the bias estimates, it allows the overall process to be faster while not preventing to compute unbiased landmarks or a drift-free vehicle pose if needed.

Concerning the data association, we use the fact that the bias model allows us to achieve consistency as a way to constrain and drive the association process. However, as consistent landmarks are not immediately available, correspondences cannot be directly sought. Instead of landmarks, we decided to work with vehicle poses. Like shown in Figure 2, a separate state vector is maintained with vehicle poses integrating the bias. Each time a new bias estimate is initialized, so is a vehicle pose in this vector. The idea is to reduce the search space around potential association areas. To do that, vehicle poses which are compatible, meaning that the vehicle could potentially be located in these already visited areas, are used. This unary constraint is computed as follows:

$$D_{ik}^2 = (\mathbf{v}_{u_i} - \mathbf{v}_{u_k})^T (\mathbf{P}_{\mathbf{v}_{u_i}} + \mathbf{P}_{\mathbf{v}_{u_k}})^{-1} (\mathbf{v}_{u_i} - \mathbf{v}_{u_k}) < \chi_{d,\alpha}^2 \quad (14)$$

where $\mathbf{v}_{u_i}$ is the vehicle pose associated to the bias $\mathbf{b}_i$ and $\mathbf{v}_{u_k}$ is the current one.

The acceptation threshold, below which an association is possible, is based on the Chi-squared distribution $\chi_{d,\alpha}^2$, with $d = 3$ (degrees of freedom) and the desired confidence α set to 0.95.

Only landmarks mapped between the different portions where vehicle poses are compatible will be tested in the association process, thus drastically reducing the number of potential pairings and the processing time required.

Once potentially already seen landmarks are extracted, they are tested with the newly mapped landmarks. To do so, we use the Geometric Constraints Branch and Bound (GCBB) method [13]. The idea is to find sets of landmarks with similar binary geometric constraints. Corresponding landmarks should share the same geometric organization. Considering two potential associations $a_{ij} = (\mathbf{l}_i, \mathbf{l}_j)$ and $a_{kl} = (\mathbf{l}_k, \mathbf{l}_l)$, a binary geometric constraint is a relation between $\mathbf{l}_i$ and $\mathbf{l}_k$ which is also satisfied between $\mathbf{l}_j$ and $\mathbf{l}_l$. In our case, $\mathbf{l}_i$ and $\mathbf{l}_k$ would be new landmarks and $\mathbf{l}_j$ and $\mathbf{l}_l$, previously mapped landmarks. The geometric constraint used here is the relative distance between the landmarks (the distance between $\mathbf{l}_i$ and $\mathbf{l}_k$ must be similar to the one between $\mathbf{l}_j$ and $\mathbf{l}_l$). Binary constraints can be computed beforehand (and are limited to landmarks already satisfying the unary constraint). A tree of the potential sets of associations is then explored to find the best correspondences.

By using the consistency provided by the bias integration, we are able to limit the exploration time needed to find associations. The use of binary geometric constraints has the advantage to allow us to find pairings between landmarks which are not affected by the same orientation drift.

With geo-referenced information, the data association differs as landmarks are not necessarily expressed with relation to the same reference (local frame or absolute frame). In this case, we only use the geometric organization of new landmarks with the GCBB algorithm to associate them with geo-referenced ones.

V. Experiments

We present two experiments to validate our approach. The first one shows a loop closure and the second one the integration of geo-referenced information. In both experiments, an electric vehicle (VIPALab) was manually driven at around 2 meters per second. Here, we do not consider the low-level SLAM to only focus on the high-level. Landmark positions and vehicles poses are received regularly without knowing what sensors or algorithms are used in the low-level (interested readers can refer to [4] for more information regarding the low-level SLAM).

The first trajectory presents a loop of approximately 100 meters. The vehicle returns to its starting point and continues

for several meters in order to have enough landmarks to detect loop closures. Throughout the trajectory, 372 landmarks were mapped, with around 40 common to both passages (around the starting point of the trajectory). We compare the proposed algorithm to its previous version [3]. In this previous version, the architecture is different and the association algorithm is based on Indivudal Compatibility between landmarks. Both Matlab implementations are checked against the same trajectory.

Figure 3 shows the trajectories performed. It can be noticed that the trajectory out of the low-level SLAM slowly drifts from the ground truth (provided by a RTK-GPS). Conversely, when integrating the bias, the algorithm is able to detect the loop and correct the trajectory. Of course, the farthest part from the closing point is the less affected by the correction. Similar results are obtained by our previous architecture, showing that integrating landmarks in the high-level state vector does not improve the localization accuracy (both trajectories are superimposed). The data association process worked well in both cases, allowing to close the loop several times around the starting point of the trajectory. For the first closure, individual compatibility between landmarks retained 160 of them (out of 372) for the GCBB process and individual compatibility between vehicle poses led to 145 landmarks for GCBB. These similar numbers show that using vehicle poses to limit the exploration tree of GCBB works as efficiently as using directly individual compatibility between landmarks. Landmarks integration inside the high-level state vector is consequently not essential to the data association process.

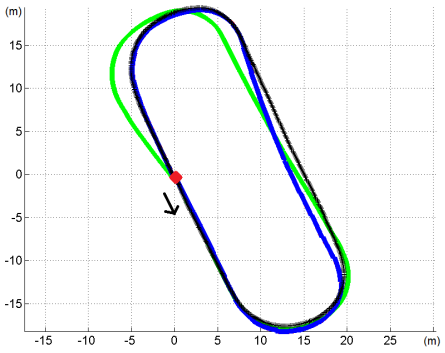


Fig. 3. Trajectories performed. In green: trajectory computed by the low-level SLAM. In blue: trajectory computed with the approach presented in this paper. In red (almost identical to the blue one): trajectory computed with the previous bias integration. In black: ground truth (RTK-GPS).

Concerning the accuracy of the obtained localization, we measured the Root-Mean-Square Error (RMSE) based on the ground truth. Results are visible in Figure 4. Closing the loop allows us to have a better position accuracy especially in the late trajectory (the beginning is quite similar).We obtain almost identical results between the approach of this paper and [3] (both curves are superimposed). The landmarks associated are the same in both methods hence the similar results. What is interesting is that the RMSE would have grown without the bias integration if the trajectory had continued because of the angular drift. This effect would have been prevented with our algorithm.

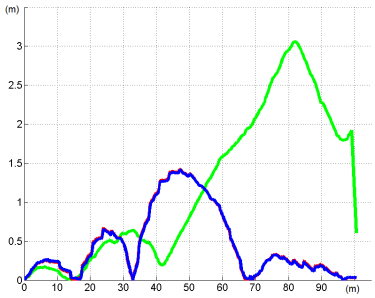
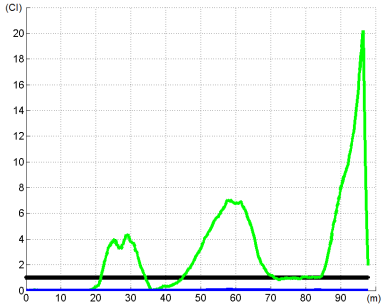
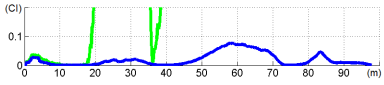


Fig. 4. Root-Mean-Square Error during the trajectory. In green: RMSE of the low-level SLAM. In blue: RMSE with the approach presented in this paper. In red (almost identical to the blue one): RMSE with the previous bias integration.

As the primary goal of the bias model is to ensure consistency, we measured the Consistency Index (CI), as defined in [14], which is based on the Normalized Estimation Error Squared (NEES) and the Chi-square test. A CI lower than 1 means that the estimate is consistent with the ground truth. Conversely, estimates whose CI is greater than 1 are optimistic (inconsistent). CI curves with no association performed are pictured in Figure 5 and in Figure 6 when the loop is closed. The bias model allows us to achieve consistency even without applying any loop closure. It is obviously not the case without the bias integration as the CI quickly grows. Once again, similar results are obtained for both architectures showing that estimating only bias values in the EKF is sufficient.



(a) Consistency Index



(b) Closer view

Fig. 5. Consistency Index during the trajectory with no association performed.In green: CI of the low-level SLAM. In blue: CI with the approach presented in this paper. In red (superimposed with blue): CI with the previous bias integration. The black line represents the consistency threshold.

The main motivation behind this architecture was to provide a faster and less memory thrifty bias integration. Figure

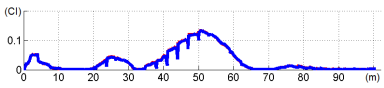
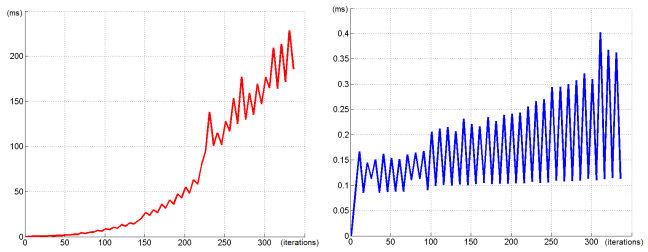


Fig. 6. Consistency Index during the trajectory with associations. In blue: CI with the approach presented in this paper. In red (superimposed with blue): CI with the previous bias integration.

7 shows the computing time taken by each algorithm all along the trajectory. The new architecture shows impressive results by using in average around 0.25 ms per iteration (integration of new landmarks and new bias estimate if needed) near the end of the trajectory while our previous implementation needs more than 220 ms for the same thing. It makes our approach suited for long trajectories. If necessary, bias estimates can easily be separated in substates and so reduce the time needed by the current architecture even more.



(a) Previous architecture (b) Current architecture

Fig. 7. Processing time required by the high-level algorithm.

In terms of memory consumption, the same trend can be noted (see Figure 8). Storing the different state vectors and covariance matrices requires almost 5 MB of memory at the end of the trajectory with the previous architecture whereas the new approach only uses around 100 KB. The exponential growth of our previous architecture is also way less pronounced with the new bias integration.

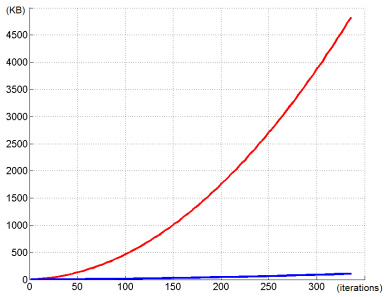


Fig. 8. Memory used by the high-level algorithm. In blue: memory used by the approach presented in this paper. In red: memory used by the previous bias integration.

In the second experiment, we show the integration of geo-referenced landmarks. The context is the same as for the previous experiment. We start the trajectory with the geo-referenced landmarks already known. The geo-referenced landmarks are created beforehand using the RTK-GPS. We generated 20 landmarks at mid-distance of the trajectory, with less than half common with the mapping process. As mentioned previously, we only use the geometric criterion to recognize these landmarks. The results can be seen in Figure 9.

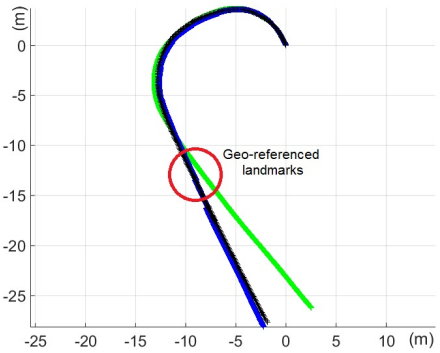


Fig. 9. Trajectories performed. In green: trajectory computed by the low-level SLAM. In blue: trajectory computed with the approach presented in this paper. Circled in red: location of the geo-referenced landmarks. In black: ground truth (RTK-GPS).

We can see that the integration of the geo-referenced landmarks allows us to correct the orientation drift that occurs during the bend. The localization is greatly improved even with only a few landmarks. The GCBB algorithm was able to find the common landmarks only using the geometric criterion (9 out of 20 were associated to the SLAM map). As previously, the RMS Error and the CI obtained by our algorithm are exposed in Figures 10 and 11 with relation to their counterpart which does not integrate the drift.

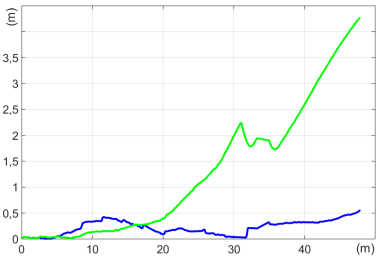


Fig. 10. Root-Mean-Square Error during the trajectory. In green: RMSE of the low-level SLAM. In blue: RMSE with the approach presented in this paper.

The RMS Error is below 50 centimeters all along the trajectory and is under 20 centimeters close to the fusion point. Without new information to characterize the drift, it starts to grow again by the end of the trajectory. Without the bias integration, the error reaches 4 meters and would have gone further if the trajectory had continued. Regarding consistency, the same behavior can be observed. During the bend and without our dedicated architecture, consistency is lost. The Covariance Index goes above a value of 300 by the end of the trajectory and keeps on growing as there is no

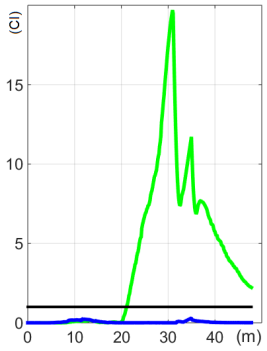


Fig. 11. Consistency Index during the trajectory with associations (zoomed view). In green: RMSE of the low-level SLAM. In blue: RMSE with the approach presented in this paper. The black line represents the consistency threshold.

way to consider the drift. With the architecture presented in this paper, the CI stays below 1 meaning that consistency is maintained throughout the whole trajectory, even when no information to estimate the drift is available.

VI. Conclusion

We have presented a new architecture allowing us to take into account the drift inherent to SLAM algorithms. Thanks to this approach, the consistency of the computed localization can be maintained even without information to correct it. The new bias integration introduced in this paper requires very little resources compared to previous approaches thus permitting to consider drift integration even with long trajectories.

The bias model is integrated outside of the classic SLAM algorithm. It has the advantage to allow the use of any feature-based SLAM algorithm. The inclusion of the localization bias in the high-level EKF-SLAM provides an easy way to estimate the drift as consistency is ensured. Absolute information, loop closing or distant information can be easily integrated. We introduced a new data association process that takes advantage of the consistency provided by the bias model. Thanks to it, it becomes possible to detect loops without relying on clues of a specific sensor. Results showed that the localization obtained is consistent and that the overall architecture allows better performance over our previous approach.

We plan to expand this framework to various scenarios using diverse SLAM algorithms. A different bias model is also envisaged to deal with non-systematic errors that cannot be covered by motion models.

References

[1] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM Algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, 2006.

[2] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 2001.

[3] G. Bresson, R. Aufrère, and R. Chapuis. Consistent Multi-robot Decentralized SLAM with Unknown Initial Positions. In *16th International Conference on Information FUSION*, 2013.

[4] G. Bresson, T. Feraud, R. Aufrere, P. Checchin, and R. Chapuis. Real-time monocular slam with low memory requirements. *Intelligent Transportation Systems, IEEE Transactions on*, PP(99):1–13, 2015.

[5] J. A. Castellanos, R. Martinez-Cantin, J. D. Tardós, and J. Neira. Robocentric Map Joining: Improving the Consistency of EKF-SLAM. *Robotics and Autonomous Systems*, 55(1):21–29, 2007.

[6] D. M. Cole and P. M. Newman. Using Laser Range Data for 3D SLAM in Outdoor Environments. In *IEEE International Conference on Robotics and Automation*, pages 1556–1563, 2006.

[7] G. Conte and P. Doherty. Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information. *EURASIP Journal On Advances In Signal Processing*, pages 10–32, 2009.

[8] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[9] C. Estrada, J. Neira, and J. D. Tardós. Hierarchical SLAM: real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[10] S. J. Julier and J. K. Uhlmann. A Counter Example to the Theory of Simultaneous Localization and Map Building. In *IEEE International Conference on Robotics and Automation*, pages 4238–4243, 2001.

[11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[12] A. Martinelli, N. Tomatis, and R. Siegwart. Some Results on SLAM and the Closing the Loop Problem. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2917–2922, 2005.

[13] J. Neira, J. D. Tardós, and J. A. Castellanos. Linear time vehicle relocation in SLAM. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 427–433, 2003.

[14] L. M. Paz, J. D. Tardós, and J. Neira. Divide and Conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5):1107–1120, 2008.

[15] S. Thrun. A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots. *The International Journal of Robotics Research*, 20(5):335–363, 2001.